Solución examen 14 de diciembre de 2017

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

Formato

- **Indique su nombre completo y número de cédula en cada hoja (No se corregirán las hojas sin nombre, sin excepciones). Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva.**
- **Si se entregan varias versiones de un problema solo se corregirá el primero de ellos.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 30 minutos del examen.**

Material

- **El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del examen, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.**

Aprobación

- **Para aprobar el examen se debe tener un mínimo de 60 puntos.**

Finalización

- **El examen dura 4 horas.**
- **Al momento de finalizar el examen no se podrá escribir absolutamente nada en las hojas, debiéndose parar e ir a la fila de entrega. Identificar cada una de las hojas con nombre, cédula y numeración forma parte de la duración del examen.**

Problema 1 (32 puntos)

Para las siguientes partes, conteste justificando brevemente, cada una de las preguntas.

1. Defina las estrategias de reemplazo de marcos: Reemplazo global y Asignación local.
2. ¿En un sistema computacional, qué permite la protección de CPU?
3. En un sistema operativo con un esquema de paginación:
   a) ¿Por qué se paginan las tablas de páginas?
   b) ¿Para qué sirve el *Translation Look-aside Buffer*?
4. En el contexto del subsistema de E/S. Describa la diferencia entre una operación no bloqueante y una asincrónica.
5. En un sistema con soporte para hilos, ¿hay un stack por hilo o un stack por proceso en el sistema? ¿y heap? Justifique y explique su respuesta.
6. Describa brevemente dos limitaciones de los hilos a nivel de usuario sin soporte del núcleo.
7. Describa los sistemas operativos diseñados con un micro-núcleo (microkernel).
8. Las estructuras RAID brindan servicios de mejora en la confiabilidad y en los tiempos de transferencias. Haga una tabla que clasifique los RAID tipo 0, 1 y 5, mencionando que servicio(s) mejoran.

9. Explicar por qué las construcciones P(s) y V(s) deben ejecutarse en forma indivisible en un semáforo no binario. Comentar el caso de semáforo binario.
10. Sea un sistema con paginación, realice un diagrama de una traducción de una dirección virtual de 32bits con 2 niveles de tabla de página. Tome en cuenta que la tabla de página de primer nivel tiene 2048 entradas y las tablas de página de segundo nivel tienen 1024 entradas.
11. Para invocar un llamado al sistema, ¿es necesario estar en modo monitor? ¿Por que?
12. Describa los hipervisores de tipo 1 y de tipo 2. Mencione una ventaja de cada uno de ellos.

Problema 2 (34 puntos)

Se desea implementar un sistema operativo con soporte multimedia. El sistema de archivos utiliza una estrategia indexada multinivel de dos niveles y un mapa de bits para administrar los bloques libres. A continuación se presentan las estructuras de datos utilizadas por este sistema de archivos.

```
const MAX_BLOQUES = 65536;
const MAX_INODOS = 8192;
type bloque = array [0..8191] of byte;          // 8192 bytes
type mapa_bits = array [0..MAX_BLOQUES-1] of bit;

type entrada_dir = Record
      usado : bool;                       // 1 bit
      nombre : array [0..123] of char;    // 124 bytes
      es_dir : bool;                      // 1 bit
      inodo_num : int16;                  // 2 bytes
      permisos : array [0..13] of bit;    // 14 bits
End; // 128 bytes

type inodo = Record
      usado: bool;                        // 1 bit
      inodo_num: int16;                   // 2 bytes
      es_dir: bool;                       // 1 bit
      es_multimedia: bool;                // 1 bit
      hLink_cant;                         // 4 bit
      tamaño: int32;                      // 4 bytes
      directo: array [0..7] of int16;     // 16 bytes
      directo_tope: int16;                // 2 bytes
      indirecto: int16;                   // 2 bytes
      indirecto_tope: int16;              // 2 bytes
      reservado: array [0..25] of bit;    // 25 bits
End; // 32 bytes
```

```
type inodos_tabla = array [0..MAX_INODOS-1] of inodo;
type disco = array [0..MAX_BLOQUES-1] of bloque;


var   IT : inodos_tabla;
      MB : mapa_bits;
      D : disco;
```

Se sabe que:

- las variables `IT`, `MB`, y `D` son globales
- en `MB` se indica con `1` si un bloque está ocupado y con `0` en caso contrario
- el inodo número `0` es el directorio raíz
- las entradas de los directorios son almacenadas utilizando un array de entradas
- `es_multimedia` indica si un archivo es multimedia
- `hLink_cant` indica la cantidad de hard links de un inodo

Se dispone de los siguientes procedimientos:

- `Procedure leerBloque(disco d, int bloque_num, byte[] &buff, bool &ok);`

  Lee desde el disco d el bloque con índice bloque_num en la variable buff. En ok se retorna verdadero si se ejecutó correctamente.
- `Procedure escribirBloque(disco d, int block_num, bytes[] buff, bool &ok);`

  Escribe en el bloque con índice bloque_num del disco d la información que se encuentra en la variable buff. En ok se retorna verdadero si se ejecutó correctamente.
- `Procedure obtenerDirPadre(char[] camino, char[] &dir);`

  Retorna en el parámetro dir el nombre del directorio padre que contiene el archivo o directorio referenciado en el parámetro camino.
  Ej: `obtenerDirPadre('/home/sistoper/a.txt') retorna '/home/sistoper'`
- `Procedure obtenerNombre(char[] camino, char[] &base);`

  Retorna en el parámetro base el nombre del archivo o directorio referenciado en el parámetro camino.

  Ej: `obtenerNombre('/home/sistoper/a.txt') retorna 'a.txt'`

Se pide:

1. Indique el tamaño máximo que puede tener un archivo utilizando el sistema de archivos planteado.
2. Implemente la siguiente función que dada una ruta, encuentre el inodo referenciado:

   `Procedure obtenerInodo(char[] ruta, int &inodo, bool &ok);`
   En `ok` se retorna verdadero si se ejecutó correctamente. En inodo se retorna el número de inodo correspondiente a la ruta.
3. Para optimizar el acceso a los archivos multimedia se utiliza una lista enlazada implementada con la variable global `IM`.

   `type inodos_mmedia_linkedlist = array [0..MAX_INODOS-1] of int16;`

```
var IM: inodos_mmedia_linkedlist;
```

Cada posición en el arreglo `IM` indica si su inodo correspondiente en `IT` contiene un inodo multimedia, y el valor almacenado en `IM` indica el siguiente inodo multimedia. El índice cero nunca referencia un archivo multimedia (porque corresponde al directorio raíz) y contiene el valor del primer inodo multimedia. El fin de la lista `IM` se indica con el valor cero.

Implemente la siguiente función que retorne todos los archivos multimedia que se encuentren en el disco:

```
Procedure obtenerMultimedia(int &tope, int[] &inodos, bool &ok);
```

En ok se retorna verdadero si se ejecutó correctamente. En inodos se retornan los números de inodo multimedia encontrados. En tope se retorna la cantidad de archivos multimeda encontrados. **Nota:** asuma que el arreglo `inodos` siempre tiene largo `MAX_INODOS`

4. Considerando la optimización planteada en la parte 3 y que el sistema soporta hard links. Implemente la siguiente función que dada una ruta, borre el archivo correspondiente:

   ```
   Procedure borrarArchivo(char[] ruta, bool &ok);
   ```

   En ok se retorna verdadero si se ejecutó correctamente.

Problema 3 (34 puntos)

Se desea modelar el arribo y embarque de pasajeros a un aeropuerto. En primera instancia los pasajeros deberán dirigirse a uno de los 10 mostradores para hacer check-in. En cada mostrador hay un empleado para atender a los pasajeros. Los pasajeros deberán elegir el mostrador con la cola más corta. Los pasajeros VIP serán atendidos con prioridad en la cola que eligieron (si hay varios VIP se atenderán por orden de llegada).

Durante el check-in recibirán un número de puerta y de asiento. Deberán dirigirse a la puerta correspondiente de las 20 que posee el aeropuerto y esperar a que se aborde el avión. Los aviones cuentan con 80 asientos y los pasajeros deberán subir en este orden: primero los asientos 61 a 80, luego 41 a 60, luego 21 a 40 y finalmente de 1 a 20. En cada puerta hay una azafata quien llamará a cada grupo de pasajeros en ese orden y ningún pasajero podrá subir antes de ser llamado. Al ser llamado el pasajero debe llegar hasta la puerta y mostrarle su ticket a la azafata. Si pasan más de 10 minutos sin que algún pasajero del grupo actual llegue a la puerta, la azafata pasará a llamar al grupo siguiente. Los pasajeros que llegan tarde no podrán abordar.

Se pide: Implementar en ADA los procesos: pasajero, empleado del mostrador y azafata

Se permite usar hasta 10 tareas auxiliares.

Se cuenta con las siguientes funciones auxiliares:

- soy_vip: boolean — Ejecutada por el pasajero para saber si es VIP.
- check_in():[puerta, asiento] — Ejecutada por el empleado del mostrador.
- llegar_a_puerta() — Ejecutada por el pasajero para ir hasta la puerta del avión cuando sea su turno de subir.
- abordar() — Llamada por el pasajero para subir al avión.
- esperar_abordaje() — Ejecutada por la azafata para esperar que el avión esté listo para abordar.
- validar_ticket():boolean — Ejecutada por la azafata para verificar que el ticket del pasajero corresponde a los asientos llamados actualmente.

Solución ejercicio 2

1)

Cada bloque es de 8192 bytes = 2^{13} bytes

Por inodo se tiene:

- 8 bloques de primer nivel (directo)
- Para referenciar un bloque se requiere de 2 bytes
  - el sistema de archivos utiliza una estrategia de indexación multinivel de dos niveles. Por lo que se cuenta con:
    - 1 bloque que referencia a bloques -> $2^{13}/2 = 2^{12}$ = 4096 bloques

Entonces: Por archivo se puede tener hasta 4096 + 8 = 4104 bloques -> 4104*8192 bytes

2)

```
procedure buscarEntradaDir(entrada_dir[] bloque, char[] nombre, int16 &entrada){
  entrada := -1;
  for(int i:= 0; i<64; i++){
    if(bloque[i].usado && bloque[i].nombre == nombre){
      entrada := i;
    }
  }
}

procedure buscarInodo(int16 inodo, char[] nombre, int &nroInodo, bool &ok){
      entrada_dir[64] bloque;
      int16[4096] bloqueInd;
      inodo in = IT[inodo]
      ok=true;
      if(!in.es_dir){
            ok = false;
      } else{
            int entry = -1;
            //busco en directos
            for(int i=0; i<in.directo_tope && entry<0 && ok; i++){
                  leerBloque(D, in.directo[i], bloque, ok);
                  if(ok){
                        buscarEntradaDir(bloque, nombre, entry)
                  }
            }
            if(ok && entry<0 && in.indirecto_tope>0){
                  leerBloque(D, in.indirecto, bloqueInd, ok);
                  if(ok){
                        for(int i=0; i<in.indirecto_tope && entry<0 && ok; i++){
                              leerBloque(D, bloqueInd[i], bloque, ok);
```

```
                              if(ok){
                                    buscarEntradaDir(bloque, nombre, entry)
                              }
                        }
                  }
            }
            if(!ok || entry<0){
                  ok=false;
            } else{
                  nroInodo=bloque[entry].inodo_num;
            }
      }
}

procedure obtenerInodo(char[] ruta, int &inodo, bool &ok){
      char[] dir;
      char[] nombre;
      obtenerDirPadre(ruta, dir);
      obtenerNombre(ruta, nombre);

      if(dir == “” and nombre == “/”){
            inodo = 0;
      } else{
            int inodoPadre;
            obtenerInodo(dir, inodoPadre, ok);
            if(ok){
                  buscarInodo(inodoPadre, nombre, inodo, ok)
            }
      }
}

3)
procedure obtenerMultimedia(int &tope, int[] &inodos, bool &ok){
      int next = IM[0];
      tope=0;
      ok=true;

      while(next>0 && next<MAX_INODOS && ok){
            if(IT[next].es_multimedia && !IT[next].es_dir){
                  inodo[tope] = next;
                  tope++;
            } else{
                  ok=false;
```

```
            }
            next=IM[next];
      }
      if(next<0 || next>MAX_INODOS){
            ok=false;
      }
}

4)
procedure borrarArchivo(char[] ruta, bool &ok){
    int inodo, inodo_p, tope, bloque_num;
    int[] inodos;
    int16[4096] bloque, bloqueInd;
    obtenerInodo(obtenerDirPadre(ruta), inodo_p, ok);
    if (!ok) return;
    if(!IT[inodo_p].es_dir){
        ok=false;
        return;
    }
    //busco en directos
    int entry = -1;
    for(int i=0; i<IT[inodo_p].directo_tope && entry<0 && ok; i++){
        bloque_num = IT[inodo_p].directo[i];
        leerBloque(D, bloque_num, bloque, ok);
        if(ok){
            buscarEntradaDir(bloque, nombre, entry)
        }
    }
    if(ok && entry<0 && IT[inodo_p].indirecto_tope>0){
        leerBloque(D, IT[inodo_p].indirecto, bloqueInd, ok);
        if(ok){
            for(int i=0; i<IT[inodo_p].indirecto_tope && entry<0 && ok; i++){
                bloque_num = bloqueInd[i];
                leerBloque(D, bloque_num, bloque, ok);
                if(ok){
                    buscarEntradaDir(bloque, nombre, entry)
                }
            }
        }
    }
    if(!ok || entry<0) {
        ok=false;
        return;
```

```
    }
    inodo = bloque[entry].inodo_num;
    bloque[entry].usado=false;
    escribirBloque(D, bloque_num, bloque, ok);

    if (!ok) return;
    IT[inodo].hLink_cant--;

    if (IT[inodo].hLink_cant == 0) {     //Si tengo que borrar
        if(IT.indirecto_tope>0){
           leerBloque(D, IT[inodo].indirecto, bloqueInd, ok);
        }
        if (!ok) return;

        //Se libera bloques directos
        for(int i=0; i<IT[inodo].directo_tope, i++){
            MB[IT[inodo].directo[i]]=0;
        }

        //Se libera bloques indirectos
        if(IT.indirecto_tope>0){
            for(int i=0;i<IT[inodo].indirecto_tope;i++){
                MB[bloqueInd[i]]=0;
            }
            MB[IT[inodo].indirecto]=0;
        }
        IT[inodo].usado=false;


        // Si el archivo es multimedia se acomoda MI
        if (IT[inodo].es_multimedia) {
            obtenerMultimedia(tope, inodos, ok);
            if(ok){
               bool fin= false;
               while(i<tope && !fin){
                  if(inodos[i]==inodo){
                     if(i==0){
                        MI[0]=MI[inodo]
                        fin=true;
                     }else{
                        MI[inodos[i-1]]=MI[inodo];
                        fin=true;
                     }
```

```
                    }
                    i++;
                }
            }
        }
    }
}
```

Solución ejercicio 3

```
task admin_mostrador
      entry llegue(out mostrador: integer);
      entry sali(mostrador: integer);
end

task body admin_mostrador
      var encolados: array(10) of integer = {0...0};
      var min, actual, n: integer;

      while true
            min = 0;
            for n:=1 to 9
                  if encolados[n] < encolados[min]
                        min = n;
                  end
            end

            select
                  accept llegue(out mostrador: integer)
                        mostrador = min;
                  end
                  encolados[min]++;
            or
                  accept sali(mostrador: integer)
                        actual = mostrador;
                  end
                  encolados[actual]--;
            end
      end
end

var empleados: array(10) of empleado;

task type empleado
      entry normal(out puerta: integer, out asiento: integer);
      entry vip(out puerta: integer, out asiento: integer);
end

task body empleado
      while true
            select
                  when vip'count == 0 =>
                  accept normal(out puerta: integer, out asiento: integer)
                        (puerta, asiento) = check_in();
                  end
```

```
            or
                  accept vip(out puerta: integer, out asiento: integer)
                        (puerta, asiento) = check_in();
                  end
            end
      end while
end empleado

var azafatas: array(20) of azafata;

task type azafata
      entry abordar[0..3] ();
      entry llegue(out ok: boolean);
end

task body azafata
      var llego_alguien: boolean;
      var i: integer;

      while true
            esperar_abordaje();

            for i := 0 to 3
                  while abordar[i]'count > 0
                        accept abordar[i]();
                  end

                  llego_alguien = true;
                  while llego_alguien
                        select
                              accept llegue(out ok:boolean)
                                    ok = validar_ticket();
                              end
                        or
                              delay 10*60; // 10 mins
                              llego_alguien = false;
                        end
                  end while
            end for
      end while
end azafata

task type pasajero
end

task body pasajero
      var mostrador, puerta, asiento: int;
      var okt: bool;

      admin_empleado.llegue(mostrador);

      if soy_vip()
            empleados[mostrador].vip(puerta, asiento);
      else
            empleados[mostrador].normal(puerta, asiento);
      end

      admin_empleado.sali(mostrador);
```

```
      if asiento >= 61
            azafatas[puerta].abordar[0]();
      else if asiento >= 41
            azafatas[puerta].abordar[1]();
      else if asiento >= 21
            azafatas[puerta].abordar[2]();
      else
            azafatas[puerta].abordar[3]();

      llegar_a_puerta();

      azafatas[puerta].llegue(okt);

      if (okt) abordar();
end pasajero
```